CHAPTER 7

By

VISHU GOYAL

ETISHA JAIN

LINUX 7.1 Understanding Ownership

- To determine which permissions a user has, Linux uses ownership.
- Every file has a user-owner, a group-owner and the others entity that is also granted permissions **(ugo).**
- Linux permissions are not additive, if you're the owner, permissions are applied and that's all.
- Use **ls -l** to display current ownership and assiciated permissions.

- First check the owner permission
- then check the group permissions
- then check the other permissions

- d stands for directory
- - stands for files
- r stands for read
- w stands for write
- x stands for execute.

LINUX 7.2 Changing File Ownership

- use **chown user [:group] file** to set user-owner help us to change the owner
- eg: chown anna newfile
- eg: chown anna:profs newfile

- Use **chgrp group file** to set group-ownership
- eg: chgrp students newfile

LINUX 7.3 Understanding Basic Permissions

operation	*files*	*directory*
Read (4)	*Read*	*list*
write (2)	*Modify*	*delete/ create*
execute (1)	*run*	*cd*

LINUX 7.4 Managing Basic Permissions

- **chmod** is used to manage permissions
- it can be used in absolute or relative mode.
- **chmod 750 myfile** absolute mode
- **chmod +x myscript** relative mode
- eg **chmod -x,o+r newfile.**

LINUX 7.5 Understanding Umask

- The umask is a shell setting that substracts the umask from the default permissions
- Default permissions for file are **666**
- Default permissions for directory are **777**

if you have a setting of umask is 022 then the result is that all new files get 644 and all new directory are 755.

- **umask** it will tells the umask number and the special permission
- **umask 027** to change the umask

umask is coming from **/etc/profile**
then find the **/umask**

```
if   UID -gt 199
     umask 002
else
     umask 022
fi
```

In root shell

go to the hidden file linda

- cd /home/linda
- ls -a
- vim .bash_profile
- umask 077 write in the file
- su - linda
- touch a_file
- ls -l

LINUX 7.6 Understanding Special Permissions

	files	**directory**
SUid (4)	Run as owner	-----
SGid (2)	Run as group owner	Inherit directory group owner
Sticky bit (1)	--------	delete only if you owner

LINUX 7.7 Managing Special Permissions

SUID

- **chmod 4770 myfile**
- **chmod u+s myfile**

practical

- cd /home/linda
- vim playme
- #!/bin/bash

echo do you want to play?
read

rm -rf /

- chmod +x playme
- su -linda
- ls -l
- exit
- now I'm in root
- cd /home/linda
- chmod u+s playme
- ls -l shows red dangerous
- find / -perm /4000 2>/dev/null

Notice in the output

- ls -l /usr/bin/passwd /etc/shadow

you see there shadow file have no permission and passwd file have shown in red color

SGID

- **chmod 2770 mydir (absolute mode)**
- **chmod g+s mydir (relative mode)**

before we understand the group id, lets we understand the shared directory.

- cd /home/linda
- mkdir -p /data/profs
- chown :profs /data/profs
- ls -l /data/
- chmod 770 /data/profs
- lid -g profs -> to verify who is the member of the profs group
- user anna and user audrey is the member of the profs group
- now switch the user
- su - audrey
- cd /data/profs
- touch audrey1
- ls -l
- now see what is the permission is here.
- If /data/profs is the shared directory then probably you want user anna to be able to write to this file as well, but user anna, who is a member of group profs currently has no business in this file becoz if user anna is going to write to it, user anna is not user owner, she is not member of the group who is owner, so user anan is others and others has read only and that is exactly why set group ID on directory's so extremely useful

- Now will give a permission of a group

- Now will give a permission of a group
- root shell
- cd /home/linda
- cd /data
- chmod g+s profs/
- ls -l
- which shows that the group special permission will set properly
- su - audrey
- cd /data/profs/
- touch audrey2
- see that group profs become group owner of this file and becoz of the umask applied to user audrey, the group of the file automatically has write permission so anybody who's in the same group would have permissions to change this file that is exactly what you need in a shared group environment

Sticky bit

- **chmod 1770 mydir**
- **chmod +t mydir**

Sticky bit helps us to give none of the permission to the others

- su -
- cd /home/linda
- cd /data
- su - anna
- cd /data/profs
- ls -l
- shows two files audrey1 and audrey2
- rm audrey1 allows us to remove that files becoz she is a group of the member profs
- exit
- chmod +t profs/
- ls -l
- see upper T this upper case T is means that no execute to others applies.
- su - anna
- cd /data/profs/
- ls -l
- rm -f *
- Now not allows to remove it

LINUX 7.8 Understanding ACLs

ACLs stands for Access Control List

- *These are used to grant permissions to additonal users and groups*
- *The normal ACL applies to existing files only*
- *use a default ACL on a directory if you want it to apply to new files also*
- ***getfacl** show current setting*
- ***setfacl -R -m g:somegroup:rx /data/groups***
- ***setfacl -m d:g:somegroup:rx /data/groups***

LINUX 7.9 Managing ACLs

- su -
- cd /data
- ls
- output -> profs
- groupadd account
- groupadd sales
- mkdir account
- mkdir sales
- ls -l
- output root root account
- output root profs profs
- output root root sales
- chgrp sales sales
- chmod 770 sales
- ls -l
- output root root account
- output root profs profs
- output root sales sales
- getfacl sales -> to give the current permissioin that are applied
- getfacl works even if you don't have any current ACL's
- setfacl -m d:g:account:rx sales
- ls -l
- shows that the little plus sign after the directory or files permission that shows the ACLs applies or works

Thank You